ELSEVIER

Theoretical Computer Science 292 (2003) 611–638

Theoretical Computer Science

www.elsevier.com/locate/tcs

Forbidding–enforcing systems

A. Ehrenfeucht[a], G. Rozenberg[b,c,*]

[a] *Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA*

[b] *Leiden Institute of Advanced Computer Science (LIACS), Leiden University, Niels Bohrweg 1, 2300 RA Leiden, Netherlands*

[c] *Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA*

Received July 2000; received in revised form October 2000; accepted December 2000

Communicated by A. Salomaa

Abstract

This paper presents a model of molecular computing that is based on two kinds of "boundary conditions": *forbidding* and *enforcing*. Forbidding conditions require that a contradictory (or conflicting) group of components (molecules) may not be present in a (molecular) system, as otherwise the system will "die". An enforcing condition requires that if a certain group of components (molecules) is present in a system, then eventually other components will be present in the system—hence such an enforcing condition models a molecular reaction. Thus the evolution of a system is determined by the enforcing conditions, but it is constraint by the forbidding conditions. Such *forbidding–enforcing systems* (*fe systems*) are investigated in this paper in the framework of strings—i.e., molecules are represented by strings. Each fe system defines a *family* of languages (rather than just one language, which is standard in formal language theory)—each language in this family presents a set of molecules that satisfy both forbidding and enforcing constraints. In this paper we investigate basic *computational* properties of fe systems operating on strings.

Keywords: Molecular computing; Theory of computation; Formal language theory

0. Introduction

DNA Computing is a new exciting development in the science of computing. As a matter of fact it is very interdisciplinary involving scientists from computer science, mathematics, molecular biology, chemistry, physics, and crystallography, among others. The generally acknowledged starting point for this area was the publication by

* Corresponding author. Tel.: +31-71-277-063; fax: +31-71-276-985.

E-mail address: rozenber@liacs.nl (G. Rozenberg).

0304-3975/03/$ - see front matter

PII: S0304-3975(01)00088-3

L. Adleman in "Science" in 1994 [1] reporting the results of an experiment on computing (a small instance of the Hamiltonian Path Problem) using DNA molecules and standard microbiological operations on them. Since then the area has developed very rapidly covering basic laboratory experiments, and the formulation and investigation of formal models.

Since DNA molecules can be modeled by strings (or double strings) over the alphabet $\{A, C, G, T\}$ of nucleotides, and various operations on DNA molecules can be modeled by operations on such strings (molecular biologists are doing this for years already), many models of DNA computing were formulated within formal language theory (see, e.g., [6]). As a matter of fact one of the main such models, splicing systems, has been formulated 7 years *before* Adleman's experiment [3]—these systems model the processing of DNA molecules by restriction enzymes.

This natural connection between DNA computing and formal language theory has turned out to be fruitful for both areas. Thus, e.g., splicing systems have generated a lot of research in formal language theory.

In this paper we formulate a model of molecular computing that also considers processing of (double) strings, but is very different from traditional systems (such as grammars) considered in formal language theory.

In a typical language theoretic model of DNA computing, e.g., splicing systems, one specifies the initial set of molecules as the set of strings which forms the *axiom set*, then one specifies the operations on these molecules as the set of *rewriting productions*, and the set of all strings generated by such a grammar (the *language* of the grammar) models the set of all molecules that can be obtained from the initial ones using the given molecular operations.

The model of forbidding–enforcing systems that we propose is *not* a grammatical model. It is based on boundary conditions rather than on rewriting by productions. We consider two types of conditions: forbidding conditions and enforcing conditions. *Forbidding conditions* are given as a family of forbidders, where each forbidder is a group of patterns which *cannot occur together* in the system, as otherwise the system will "die" (e.g., will lose its functionality). *Enforcing conditions* are given as a family of enforcers, where each enforcer says that if a certain group of strings (molecules) is present in the system, then some other strings (molecules) will eventually be present in the system. In this way an enforcer models a molecular reaction.

Then in a *forbidding–enforcing system*, *fe system* for short, which is specified by a set of forbidding conditions $\mathscr{F}$ and a set of enforcing conditions $\mathscr{E}$, the evolution of the system proceeds according to the molecular reactions specified by $\mathscr{E}$, *but* it is constrained by $\mathscr{F}$: the evolution cannot lead to any group of patterns specified by a forbidder from $\mathscr{F}$. In this way a fe system specifies a (possibly infinite) *family of languages* with each language obeying both $\mathscr{F}$ and $\mathscr{E}$. This is in sharp contrast to grammars considered in formal language theory, where each grammar specifies *one language*.

We believe that fe systems are novel and interesting also from the formal language theory point of view because of the above difference with traditional grammars, but

also because they are essentially different from the point of view of the methodology of defining languages. The standard constructs of formal language theory, such as grammars and automata, follow the axiom: "*everything that is not allowed is forbidden*". Thus a string w belongs to the language of a grammar (or an automaton) only if it is "explicitly" generated (or accepted)—otherwise w is forbidden (it does not belong to the language). On the other hand fe systems follow the "orthogonal" axiom: "*everything that is not forbidden is allowed*".

The paper is organized as follows.

Forbidding conditions are formalized as forbidding sets in Section 1, where we also give some basic properties of them. Enforcing conditions are formalized as enforcing sets in Section 2. Section 3 introduces a key notion of the theory of fe systems, viz. evolving through enforcing—it is formalized through the notion of $\mathscr{E}$-extension where $\mathscr{E}$ is an enforcing set.

The issue of finiteness is always important from both the theoretical and the "real world" applications point of view. We investigate in this paper many facets of finiteness in the framework of forbidding and enforcing. In Section 4 we investigate the relationship between "syntactic finiteness" and "semantic finiteness" formalized through the notions of finitary enforcing sets and weakly finitary enforcing sets, respectively. Then in Section 5 we prove that finitary fe systems are "universal" in the sense that any family of languages that can be specified by an enforcing set can also be specified by a finitary enforcing set.

In Section 6 we combine forbidding and enforcing sets and define in this way forbidding–enforcing systems. Then in Section 7 we illustrate the use of fe systems in formalizing the satisfiability problem for Boolean formulas, while in Section 8 we use fe systems to define the structure of DNA molecules and various operations on them. In Section 9 we demonstrate that finitary fe systems are intrinsically computational. Each such fe system Γ can be completely represented by a finitely branching tree which describes through the node labels all finite languages in the family of languages defined by Γ, and through the paths all evolving computations of the system, as well as all infinite languages defined by Γ.

0.1. Preliminaries

We assume the reader to be familiar with basic notions concerning sets, words, languages and trees. We recall some of them in this section in order to fix the basic terminology and notation.

For a set Z, 2^Z denotes the set of all subsets of Z, and if Z is finite, then $|Z|$ denotes its cardinality. The empty set is denoted by $\emptyset$, and for sets Z_1, Z_2, we use $Z_1 \cap Z_2$, $Z_1 \cup Z_2$ and $Z_1 - Z_2$ to denote their union, intersection and difference, respectively. Also $Z_1 \subseteq Z_2$ denotes the inclusion of Z_1 in Z_2, $Z_1 \subset Z_2$ denotes the strict inclusion of Z_1 in Z_2, while $Z_1 \not\subseteq Z_2$ says that Z_1 is not included in Z_2. We use $\mathbb{N}$ to denote the set of nonnegative integers, and for a finite subset I of $\mathbb{N}$, $\max I$ denotes the maximal integer in I.

For a binary relation R, we use $R^{(1)}$ and $R^{(2)}$ to denote the projection of R on the first and on the second coordinate, respectively, i.e., $R^{(1)} = \{x : (x, y) \in R\}$ and $R^{(2)} = \{y : (x, y) \in R\}$.

We will consider only finite alphabets, and for an alphabet Σ we will use Σ^+ and Σ^* to denote the set of all nonempty words and the set of all words over Σ, respectively. For a word w, $|w|$ denotes the length of w. For $b \in \Sigma$ and $n \in \mathbb{N}$, b^n is the word consisting of b catenated with itself n times; for $n = 0$, b^0 is the empty word.

For words u and w, we say that u is a subword of w, written $u\, sub\, w$, iff there exist words x and y such that $w = xuy$. We use $sub(w)$ to denote the set of all subwords of w, and for a language K, $sub(K) = \{x : x \in sub(w) \text{ for some } w \in K\}$.

A sequence of languages $\sigma = K_0, K_1, \ldots$ is called *ascending* if $K_i \subseteq K_{i+1}$ for all i, if σ is infinite, and $K_i \subseteq K_{i+1}$ for all $i < m$, if σ is finite and K_m is the last element of σ. Also $\bigcup \sigma$ denotes the union of all languages in σ, and $|\sigma|$ denotes the *length* of σ: if $\sigma = K_0, K_1, \ldots, K_m$ for some $m \geqslant 0$, then $|\sigma| = m + 1$, and if σ is infinite, then $|\sigma|$ equals the cardinality of the set of natural numbers.

In this paper, by a *tree* we mean a (nonempty) rooted directed node-labelled tree such that each node has only a finite number of direct descendants (including zero if the node is a leaf).

A tree τ is specified by a 3-tuple (V, E, φ) where V is the set of *nodes* of τ, E the set of *edges*, and φ is the *node-labelling function*. We also use $nd(\tau)$, $ed(\tau)$, and lab_τ to denote V, E, and φ, respectively. For a node $v \in V$, $ddes_\tau(v)$ is the set of all direct descendants of v in τ. We use $<_\tau$ to denote the transitive closure of the $ddes_\tau$ relation, i.e., for nodes v and u, $v <_\tau u$ iff there exists a sequence of nodes $v_1, \ldots, v_n$ for some $n \geqslant 2$ such that $v_1 = v$, $v_n = u$, and $v_{i+1} \in ddes_\tau(v_i)$ for all $1 \leqslant i \leqslant n - 1$.

A *path* is a sequence of nodes $\pi = v_1, v_2, \ldots$ such that either $v_{i+1} \in ddes_\tau(v_i)$ for all $1 \leqslant i < n$, and π consists of n nodes (we say then that π is a path *from* v_1 *to* v_n), or $v_{i+1} \in ddes_\tau(v_i)$ for all $i \geqslant 1$, and π is infinite. If v_1 is the root of τ, then π is a *rooted path*. If π is rooted, then π is a *complete path* if either π is finite, $\pi = v_1, \ldots, v_n$ and v_n is a leaf, or π is infinite. For a path $\pi = v_1, v_2, \ldots$ we use $\varphi(\pi)$ to denote the corresponding sequence of labels $\varphi(v_1), \varphi(v_2), \ldots$.

1. Forbidding sets

In this section we formalize one of our key notions, viz., the notion of forbidding. To this aim we introduce the formal notion of a forbidding set.

Definition 1.1. A *forbidding set* is a family of finite nonempty subsets of Σ^+ for some alphabet Σ; each element of a forbidding set is called a *forbidder*.

Note that a forbidding set may be infinite—we only require that each forbidder is finite.

Example 1.1. For $\Sigma = \{a, b\}$, the set $\mathscr{F} = \{\{ab, ba\}, \{aa, bb\}\}$ is a forbidding set. Hence $\{ab, ba\}$ and $\{aa, bb\}$ are forbidders.

Definition 1.2. Let $\mathscr{F}$ be a forbidding set, and let K be a language.

(1) Let $F \in \mathscr{F}$. We say that K is *consistent with* F, written $K \, con \, F$, iff $F \not\subseteq sub(K)$.
(2) We say that K is *consistent with* $\mathscr{F}$, written $K \, con \, \mathscr{F}$, iff $K \, con \, F$ for each $F \in \mathscr{F}$.

Example 1.2. Consider the forbidding set $\mathscr{F}$ from Example 1.1., and let $K \subseteq \{a, b\}^+$. Then it is easily seen that $K \, con \, \mathscr{F}$ iff $K \subseteq K_i$ for some $i \in \{1, 2, 3, 4\}$, where

$$K_1 = \{a\}\{b\}^* \cup \{b\}^+,$$

$$K_2 = \{a\}^*\{b\} \cup \{a\}^+,$$

$$K_3 = \{b\}\{a\}^* \cup \{a\}^+,$$

$$K_4 = \{b\}^*\{a\} \cup \{b\}^+.$$

If K is not consistent with $\mathscr{F}$, then we write $K \, ncon \, F$.

Also, $\mathscr{L}(\mathscr{F})$ denotes the class of all languages consistent with $\mathscr{F}$. Here we assume that some alphabet Σ is fixed in the context of our considerations, and $\mathscr{L}(\mathscr{F})$ is the class of all languages over Σ which are consistent with $\mathscr{F}$. If necessary, one can incorporate Σ into this notation by writing $\mathscr{L}_\Sigma(\mathscr{F})$. Also, for an alphabet Σ, we say that $\mathscr{F}$ *is over* Σ iff $F \subseteq \Sigma^+$ for each $F \in \mathscr{F}$; the minimal (w.r.t. $\subseteq$) alphabet Σ satisfying this condition is called *the alphabet of* $\mathscr{F}$. Then, a *forbidding system* is an ordered pair $(\Sigma, \mathscr{F})$ such that $\mathscr{F}$ is a forbidding set over Σ.

The following theorem gives three very basic properties of the consistency relation.

Theorem 1.1. *Let $\mathscr{F}$ be a forbidding set, and let K be a language such that $K \, con \, \mathscr{F}$.*
(1) *$sub(K) \, con \, \mathscr{F}$.*
(2) *If $K' \subseteq K$, then $K' \, con \, \mathscr{F}$.*
(3) *If $\sigma = K_1, K_2, \ldots$ is an ascending sequence of languages such that, for each $1 \leqslant i \leqslant |\sigma|$, $K_i \, con \, \mathscr{F}$, then $\bigcup \sigma \, con \, \mathscr{F}$.*

Proof. (1) and (2) follow directly from the definition of consistency.

(3) Assume that a sequence of languages σ satisfies the assumptions of statement (3) above. Let $K = \bigcup_{1 \leqslant i \leqslant |\sigma|} K_i$. If σ is finite, then the conclusion of (3) obviously holds (we have then $K = K_{|\sigma|}$).

Assume then that σ is infinite, and assume to the contrary that $K \, ncon \, F$.

Hence, there is a forbidder $F \in \mathscr{F}$ such that $F \subseteq sub(K)$. Since each forbidder is a finite language, this implies that $F \subseteq sub(K_i)$ for some $i \geqslant 1$ which contradicts the assumption that $K_i \, con \, \mathscr{F}$. Thus $K \, con \, \mathscr{F}$. $\square$

Since in this paper we are mainly interested in using forbidding sets to define language families, the following notion of equivalence is very natural.

Definition 1.3. Forbidding sets $\mathscr{F}_1$, $\mathscr{F}_2$ are *equivalent*, denoted $\mathscr{F}_1 \sim \mathscr{F}_2$, iff $\mathscr{L}(\mathscr{F}_1) = \mathscr{L}(\mathscr{F}_2)$.

2. Enforcing sets

The other key notion of this paper is the notion of enforcing—it is formalized through enforcing sets.

Definition 2.1. An *enforcing set* is a family of ordered pairs (X, Y) such that for some alphabet Σ all $X, Y \subseteq \Sigma^+$, X, Y are finite, and each $Y \neq \emptyset$; each element of an enforcing set is called an *enforcer*.

Example 2.1. Let Σ be an alphabet. The family $\mathscr{E} = \{(X, Y) : X = \{u, v\}, Y = \{uv, vu\}$ with $u, v \in \Sigma^+\}$ is an enforcing set.

Definition 2.2. Let $\mathscr{E}$ be an enforcing set, and let K be a language.

(1) Let $E = (X, Y) \in \mathscr{E}$. We say that E is *applicable* to K (or that E is K-*applicable*), written $E \, app \, K$, iff $X \subseteq K$. Then we say that K *satisfies* E, written $K \, sat \, E$, iff $E \, app \, K$ implies $Y \cap K \neq \emptyset$. If $E \, app \, K$ but $Y \cap K = \emptyset$, then E is a K-*violator*.

(2) We say that K satisfies $\mathscr{E}$, written $K \, sat \, \mathscr{E}$, iff $K \, sat \, E$ for each $E \in \mathscr{E}$.

We write $K \, nsat \, \mathscr{E}$ iff K does not satisfy $\mathscr{E}$.

Also, $\mathscr{L}(\mathscr{E})$ denotes the class of all languages satisfying $\mathscr{E}$. Again, as it was the case for forbidding sets, we assume that some alphabet Σ is fixed in the context of our considerations; otherwise we write explicitely $\mathscr{L}_\Sigma(\mathscr{E})$. Also, for an alphabet Σ, we say that $\mathscr{E}$ *is over* Σ iff $X, Y \subseteq \Sigma^+$ for each $(X, Y) \in \mathscr{E}$; the minimal (w.r.t. $\subseteq$) alphabet Σ satisfying this condition is called *the alphabet of* $\mathscr{E}$. Then, an *enforcing system* is an ordered pair $(\Sigma, \mathscr{E})$ such that $\mathscr{E}$ is an enforcing set over Σ.

Example 2.2. Let $\mathscr{E}$ be the enforcing set from Example 2.1. If $K \subseteq \Sigma^+$ satisfies $\mathscr{E}$, then K is closed under "weak catenation": for any words $u, v \in K$ at least one of the words uv, vu is in K.

Example 2.3. Let $\mathscr{E} = \{(\emptyset, \{b^n\}) : n \text{ is even}\}$. If K satisfies $\mathscr{E}$, then K must contain the language $\{b^n : n \text{ is even}\}$.

The enforcers from Example 2.3 are of a very special form, viz. (X, Y) with $X = \emptyset$. Since $\emptyset \subseteq Z$ for every set Z, such an enforcer is *always* applicable, and so if it is included in an enforcing set $\mathscr{E}$ and $K \, sat \, \mathscr{E}$, then $Y \cap K \neq \emptyset$. For this reason we refer to such enforcers as *brute enforcers*. Again (as it was the case for forbidding sets) the following notion of equivalence for enforcing sets is natural for our purposes.

Definition 2.3. Enforcing sets $\mathscr{E}_1$ and $\mathscr{E}_2$ are *equivalent*, written $\mathscr{E}_1 \sim \mathscr{E}_2$, iff $\mathscr{L}(\mathscr{E}_1) = \mathscr{L}(\mathscr{E}_2)$.

3. Evolving through enforcing

A system obeying the enforcing conditions described by $\mathscr{E}$ will exhibit some "consequential behaviour": if some things (e.g., molecules) are present, then eventually other things (molecules) must also be present. In this way enforcing dictates certain *evolving rules* for the system. This evolving through enforcing is formalized in this section. First, we need some additional terminology and notation.

Let $\mathscr{E}$ be an enforcing set.

Assume that K *nsat* $\mathscr{E}$.

Let $V_0 = \{E \in \mathscr{E} : E \text{ is a } K\text{-violator}\}$; since K *nsat* $\mathscr{E}$, $V_0 \neq \emptyset$. Let then Z_0 be a set such that for each $W \in V_0^{(2)}$, $|Z_0 \cap W| \geqslant 1$. Thus Z_0 contains at least one element from Y for each $(X, Y) \in V_0$.

Let then $K_0 = K$ and $K_1 = K_0 \cup Z_0$.

Clearly, none of the enforcers in V_0 is a K_1-violator. However, $K_0 \subset K_1$ and so $\mathscr{E}$ may contain enforcers that are K_1-applicable, but not K_0-applicable. If this is the case, then we let $V_1 = \{E \in \mathscr{E} : E \text{ is a } K_1\text{-violator}\}$. Then again we let Z_1 be a set such that $|Z_1 \cap W| \geqslant 1$ for each $W \in V_1^{(2)}$ (thus Z_1 contains at least one element from Y for each $(X, Y) \in V_1$).

Let then $K_2 = K_1 \cup Z_1$.

Again, none of the enforcers in V_1 is a K_2-violator, however $K_1 \subset K_2$ and so $\mathscr{E}$ may contain enforcers that are K_2-applicable, but not K_1-applicable.

If this is the case, then we iterate the procedure. This iterative "repair procedure" provides the intuition behind the following key notion.

Definition 3.1. For an enforcing set $\mathscr{E}$ and languages K_1, K_2 we say that K_2 is an *$\mathscr{E}$-extension* of K_1, written $K_1 \vdash_{\mathscr{E}} K_2$, iff, for each $(X, Y) \in \mathscr{E}$, $X \subseteq K_1$ implies $K_2 \cap Y \neq \emptyset$.

It follows directly from the above definition that

(1) if K *sat* $\mathscr{E}$, then $K \vdash_{\mathscr{E}} K$, and

(2) if $K_1 \vdash_{\mathscr{E}} K_2$, $K_2 \vdash_{\mathscr{E}} K_3, K_1 \subseteq K_2$ and $K_2 \subseteq K_3$, then $K_1 \vdash_{\mathscr{E}} K_3$.

Example 3.1. Let $\mathscr{E}$ be the enforcing set from Example 2.1., and let

$$K = \{a, ab, b^2\},$$

$$K_1 = \{a^2b, ab^2, ab^3, a^2, abab, b^4\} \cup K,$$

$$K_2 = \{aba, b^2a, b^2ab, a^2, abab, b^4\},$$

$$K_3 = K_2 \cup \{a, b^2\}, K_4 = K_2 - \{aba\}.$$

Note that K_1 is a closure of K under weak catenation, and K_2 contains exactly one catenation for each pair $\{u,v\}$ of words from K (including the catenation uu, when $u=v$). We have $K \vdash_{\mathscr{E}} K_1$, $K \vdash_{\mathscr{E}} K_2$, and since $K_2 \subseteq K_3$, also $K \vdash_{\mathscr{E}} K_3$.

On the other hand, K_4 contains neither a^2b nor aba, while $\{a, ab\} \subseteq K$. Hence K_4 does not contain any catenation of the words a and b, and so K_4 is not an $\mathscr{E}$-extension of K. □

The following result says that the above described repair procedure gives the desired effect.

Theorem 3.1. *Let $\mathscr{E}$ be an enforcing set and let $\sigma = K_1, K_2, \ldots$ be an infinite ascending sequence of languages. If, for each $i \geqslant 1$, $K_i \vdash_{\mathscr{E}} K_{i+1}$, then $\bigcup \sigma$ sat $\mathscr{E}$.*

Proof. Let $K = \bigcup \sigma$ and let $E \in \mathscr{E}$ be applicable to K. Hence, if $E = (X, Y)$, then $X \subseteq K$. Since X is finite, $X \subseteq K_m$ for some $m \geqslant 1$. But, because $K_m \vdash_{\mathscr{E}} K_{m+1}$, $Y \cap K_{m+1} \neq \emptyset$. Thus $Y \cap K \neq \emptyset$. Hence K *sat* E, and since E was an arbitrary enforcer from $\mathscr{E}$, K *sat* $\mathscr{E}$. □

Clearly if $K_1 \vdash_{\mathscr{E}} K_2$, then also $K_1 \vdash_{\mathscr{E}} K_2'$ for each K_2' such that $K_2 \subseteq K_2'$. Thus, in general, K_2 may contain a lot of "redundancy". Therefore one is often interested in minimal $\mathscr{E}$-extensions of a given language K_1.

Definition 3.2. For an enforcing set $\mathscr{E}$ and languages K_1, K_2 we say that K_2 is a *minimal $\mathscr{E}$-extension* of K_1 iff $K_1 \vdash_{\mathscr{E}} K_2$ and for each strict subset K of K_2 it is not true that $K_1 \vdash_{\mathscr{E}} K$.

If K_2 is a minimal $\mathscr{E}$-extension of K_1, then we also say that K_2 is $(K_1, \vdash_{\mathscr{E}})$-*minimal.*

Example 3.2. Let $\mathscr{E}$ be the enforcing set from Example 2.1, and let K, K_1, K_2, K_3 be the languages from Example 3.1. Then K_1, K_2, and K_3 are $\mathscr{E}$-extensions of K, but *only* K_2 is a minimal $\mathscr{E}$-extension of K. □

4. Finiteness conditions

The issue of finiteness is always important from both the theoretical and the "real world" applications point of view. In this section we will consider two kinds of finiteness relevant for enforcing sets.

The following notation will be used in the sequel.

For a finite language Z, $\mathscr{E}(Z) = \{(X, Y) \in \mathscr{E} : X = Z\}$ and $\bar{\mathscr{E}}(Z) = \{Y : (X, Y) \in \mathscr{E}(Z)\}$; thus $\bar{\mathscr{E}}(Z)$ is a simpler notation for $(\mathscr{E}(Z))^{(2)}$. If $\mathscr{E}(Z) \neq \emptyset$, then we say that Z is *relevant for* $\mathscr{E}$, thus $\mathscr{E}^{(1)}$ is the family of sets relevant for $\mathscr{E}$.

Definition 4.1. (1) An enforcing set $\mathscr{E}$ is *finitary*, iff, for each finite language Z, $\mathscr{E}(Z)$ is finite.

(2) An enforcing set $\mathscr{E}$ is *weakly finitary*, iff, for each finite language K_1 there exists a finite language K_2 such that $K_1 \vdash_{\mathscr{E}} K_2$.

Being finitary is a *syntactic* property, quite convenient if we have to either specify or analyze $\mathscr{E}(Z)$ for some Z relevant for $\mathscr{E}$. Being "weakly finitary" is a *semantic* property—it says that if $\mathscr{E}$ is weakly finitary, then each finite set can evolve (according to $\mathscr{E}$) into a finite set. The basic relationship between these properties is given by the following result.

Theorem 4.1. (1) *Every finitary enforcing set $\mathscr{E}$ is weakly finitary.*
(2) *There exist weakly finitary enforcing sets that are not finitary.*

Proof. (1) Let $\mathscr{E}$ be a finitary enforcing set, and let K be a finite language. Since K is finite, $\mathscr{K} = 2^K$ is finite. Since $\mathscr{E}$ is finitary, $\mathscr{E}(Z)$ is finite for each $Z \in \mathscr{K}$. Thus $M = \bigcup_{Z \in \mathscr{K}} \bar{\mathscr{E}}(Z)$ is finite, and consequently $K \cup M$ is finite. Obviously $K \vdash_{\mathscr{E}} K \cup M$. Consequently, $\mathscr{E}$ is weakly finitary.

(2) Consider $\mathscr{E} = \{(\emptyset, \{a, b^n\}) : n \geqslant 1\}$. Since $\mathscr{E}(\emptyset)$ is infinite, $\mathscr{E}$ is not finitary. Consider now a finite language K. Obviously, $K \vdash_{\mathscr{E}} K \cup \{a\}$, and so $\mathscr{E}$ is weakly finitary. □

Our next result says that if a language K satisfies a finitary $\mathscr{E}$, then K constitutes a universe (like, e.g., Σ^+ does) for finite sets to evolve into finite sets.

Theorem 4.2. *Let $\mathscr{E}$ be a finitary enforcing set, and let K be a language such that K sat $\mathscr{E}$. For every finite language $L \subseteq K$, there exists a finite language L' such that $L \subseteq L' \subseteq K$ and $L \vdash_{\mathscr{E}} L'$.*

Proof. Let $L \subseteq K$ be a finite language. Then also $\mathscr{L} = 2^L$ is finite. Since $\mathscr{E}$ is finitary, $\mathscr{E}(Z)$ is finite for each $Z \in \mathscr{L}$. Hence $M = \bigcup_{Z \in \mathscr{L}} \bar{\mathscr{E}}(Z)$ is finite, and so $M \cap K$ is finite. But $M \cap K \subseteq K$ and so $L' = L \cup (M \cap K)$ is finite and included in K. Since obviously $L \subseteq L' \subseteq K$ and $L \vdash_{\mathscr{E}} L'$, the theorem holds (because K *sat* $\mathscr{E}$). □

Note that the above result does not hold if we require that $\mathscr{E}$ is weakly finitary rather than finitary. To see this consider the weakly finitary enforcing set $\mathscr{E} = \{(\emptyset, \{a, b^n\}) : n \geqslant 1\}$ from the proof of Theorem 4.1. Let $K = \{b^n : n \geqslant 1\}$; obviously K *sat* $\mathscr{E}$. Now let $L \subseteq K$ be finite and assume that $L \vdash_{\mathscr{E}} L'$ for a finite language L'. Let $m = \max\{n : b^n \in L'\}$ and consider the enforcer $E = (\emptyset, \{a, b^{m+1}\})$. Obviously L' *nsat* E, contradicting $L \vdash_{\mathscr{E}} L'$. Hence L' cannot be finite.

5. Finitary normal form

In this section we prove that, up to equivalence $\sim$, one can consider only finitary enforcing sets. First we prove a useful normal form.

Lemma 5.1. *Let $\mathscr{E}$ be an enforcing set, and let $(X,Y_1),(X,Y_2)\in\mathscr{E}$ be such that $Y_1\subset Y_2$. Let $\mathscr{E}'=\mathscr{E}-\{(X,Y_2)\}$. Then $\mathscr{E}\sim\mathscr{E}'$.*

Proof. (1) First we note that if $\mathscr{E}_1$, $\mathscr{E}_2$ are enforcing sets such that $\mathscr{E}_1\subseteq\mathscr{E}_2$, then, for every language K, K *sat* $\mathscr{E}_2$ implies K *sat* $\mathscr{E}_1$.

(2) Let then $\mathscr{E},\mathscr{E}'$ be as in the statement of the lemma.

By (1) we have $\mathscr{L}(\mathscr{E})\subseteq\mathscr{L}(\mathscr{E}')$.

Now let $K\in\mathscr{L}(\mathscr{E}')$.

Consider now $(U,V)\in\mathscr{E}$ such that (U,V) *app* K.

If $(U,V)\neq(X,Y_2)$, then K *sat* (U,V) because $K\in\mathscr{L}(\mathscr{E}')$.

Let then $(U,V)=(X,Y_2)$.

Since $(X,Y_1)\in\mathscr{E}'$ and $K\in\mathscr{L}(\mathscr{E}')$, $K\cap Y_1\neq\emptyset$. But $Y_1\subset Y_2$, and so $K\cap Y_2\neq\emptyset$. Thus K *sat* (X,Y_2), and so K *sat* (U,V).

Consequently K *sat* $\mathscr{E}$, and so $K\in\mathscr{L}(\mathscr{E})$. Thus $\mathscr{L}(\mathscr{E}')\subseteq\mathscr{L}(\mathscr{E})$.

Hence $\mathscr{L}(\mathscr{E}')=\mathscr{L}(\mathscr{E})$ and so $\mathscr{E}\sim\mathscr{E}'$. □

Theorem 5.1. *For every enforcing set $\mathscr{E}$ there exists a finitary enforcing set $\mathscr{E}'$ such that $\mathscr{E}\sim\mathscr{E}'$.*

Proof. Let $\mathscr{E}$ be an enforcing set. By Lemma 5.1 we can assume that if $(X,Y_1),(X,Y_2)\in\mathscr{E}$ for some sets X,Y_1,Y_2, then neither $Y_1\subseteq Y_2$ nor $Y_2\subseteq Y_1$.

In order to define $\mathscr{E}'$ we will first construct for every $Z\in\mathscr{E}^{(1)}$ and auxiliary set $\mathscr{A}(Z)$ by considering separately two cases.

Case 1: $\mathscr{E}(Z)$ is finite.

Then $\mathscr{A}(Z)=\mathscr{E}(Z)$.

Case 2: $\mathscr{E}(Z)$ is infinite.

Let then $\vec{\mathscr{E}}(Z)=(Z,Y_1),(Z,Y_2),\ldots,(Z,Y_n),\ldots$ be an arbitrary but fixed ordering of $\mathscr{E}(Z)$. For example, we note that, for each $(Z,Y)\in\mathscr{E}(Z)$, Y is a finite set. We may then assume that the alphabet of $\mathscr{E}$ is ordered, and then all Y's are ordered lexicographically yielding the order $Y_1,Y_2,\ldots,Y_n,\ldots$.

Let W be a finite set and let $n>1$. We say that W is *n-critical* (for $\vec{\mathscr{E}}(Z)$) iff

(1) for each $1\leqslant i<n$, $W\cap Y_i\neq\emptyset$ and $W\cap Y_n=\emptyset$, and

(2) W is a minimal set (w.r.t. $\subseteq$) satisfying (1) meaning that no proper subset of W satisfies (1).

Clearly, our assumption about $\mathscr{E}$ guarantees that for each $n>1$ there is an n-critical set.

Let now, for each $n>1$, $W_{1,n},W_{2,n},\ldots,W_{t_n,n}$ be all n-critical sets for $\vec{\mathscr{E}}(Z)$.

Then we define $\mathscr{A}(Z)$ to be the set consisting of (Z,Y_1) and of $(Z\cup W_{1,n},Y_n)$, $(Z\cup W_{2,n},Y_n),\ldots,(Z\cup W_{t_n,n},Y_n)$ for each $n>1$.

Note that because of the minimality condition for n-critical sets, all of $W_{1,n},\ldots,W_{t_n,n}$ are included in $\bigcup_{1\leqslant j<n}Y_j$. Consequently there are only finitely many n-critical sets for each $n>1$. This justifies our index t_n.

Now we define $\mathscr{E}'$ through $\mathscr{A}$ as follows: $\mathscr{E}'=\bigcup_{Z\in\mathscr{E}^{(1)}}\mathscr{A}'_Z$.

Claim 1. *For every language K, if K sat $\mathscr{E}$, then K sat $\mathscr{E}'$.*

Proof of Claim 1. Assume that K *sat* $\mathscr{E}$. Let $(X', Y') \in \mathscr{E}'$ be applicable to K.

From the construction of $\mathscr{E}'$ it follows that there exists an $X \subseteq X'$ such that $(X, Y') \in \mathscr{E}$. Since $X \subseteq X'$, (X, Y') is applicable to K. Since K *sat* $\mathscr{E}$, $Y' \cap K \neq \emptyset$, and so K *sat* (X', Y'). Since (X', Y') was an arbitrary enforcer from $\mathscr{E}'$ applicable to K, K *sat* $\mathscr{E}'$. □

Claim 2. *For every language K, if K sat $\mathscr{E}'$, then K sat $\mathscr{E}$.*

Proof of Claim 2. Assume that K *sat* $\mathscr{E}'$.

Let $(X, Y) \in \mathscr{E}$ be applicable to K.

We consider two cases.

Case 1: $\mathscr{E}(X)$ is finite.

Then $(X, Y) \in \mathscr{E}'$. Since K *sat* $\mathscr{E}'$, K *sat* (X, Y).

Case 2: $\mathscr{E}(X)$ is infinite.

Consider the (lexicographic) ordering $\vec{\mathscr{E}}(X) = (X, Y_1), (X, Y_2), \ldots$, as used in defining $\mathscr{A}$ above.

We will prove now by induction on m that:

$$\text{for each } m \geqslant 1, \quad Y_m \cap K \neq \emptyset. \tag{1}$$

Base: $m = 1$.

Since $(X, Y_1) \in \mathscr{E}'$ and (X, Y_1) is applicable to K, $Y_1 \cap K \neq \emptyset$.

Inductive assumption: assume that $m > 1$ and that $Y_i \cap K \neq \emptyset$ for each $i < m$.

Inductive step:

First we note that there exists a $Z \subseteq K$ such that:

$$\text{for each } i < m, \quad Y_i \cap Z \neq \emptyset. \tag{2}$$

This holds because by the inductive assumption $Y_i \cap K \neq \emptyset$ for each $i < m$, and so if for each $i < m$ we choose one y_i from $Y_i \cap K$, and then we set $Z = \{y_1, y_2, \ldots, y_{m-1}\}$, then $Z \subseteq K$ and Z satisfies (2).

Let Z_0 be a minimal $Z \subseteq K$ satisfying (2), i.e., no strict subset of Z_0 satisfies (2).

If $Z_0 \cap Y_m \neq \emptyset$, then since $Z_0 \subseteq K$, $Y_m \cap K \neq \emptyset$.

If $Z_0 \cap Y_m = \emptyset$, then Z_0 is m-critical for $\vec{\mathscr{E}}(X)$.

Thus $(Z_0 \cup X, Y_m) \in \mathscr{E}'$. Since K *sat* $\mathscr{E}'$ and $Z_0 \cup X \subseteq K$, we get $Y_m \cap K \neq \emptyset$.

This completes the proof of (1).

But for some $n \geqslant 1$, $Y = Y_n$, and so by (1), $Y \cap K \neq \emptyset$. Thus K *sat* (X, Y).

Thus in both Cases 1 and 2, K *sat* (X, Y).

Since (X, Y) was an arbitrary enforcer from $\mathscr{E}$, this implies that K *sat* $\mathscr{E}$. □

Claim 3. $\mathscr{E} \sim \mathscr{E}'$.

Proof of Claim 3. Directly from Claims 1 and 2. □

Claim 4. *$\mathscr{E}'$ is finitary.*

Proof of Claim 4. Let X be relevant for $\mathscr{E}'$.

If $\mathscr{E}(X)$ is finite, then $\mathscr{A}(X)=\mathscr{E}(X)$ adds only a finite number of enforcers (X,Y), for some Y, to $\mathscr{E}'$.

Other enforcers of the form (X,Y) in $\mathscr{E}'$ may be contributed by the sets $\mathscr{A}(Z)$ for some $Z\subseteq X$ such that $\mathscr{E}(Z)$ is infinite and $(Z,Y)\in\mathscr{E}$. If this is the case, then $X=Z\cup W$ where W is a n-critical set for $\vec{\mathscr{E}}(Z)$ for some $n\geqslant 1$. Since X is finite and $Z\subseteq X$, there is only a finite number of such Z. Hence to prove that $\mathscr{E}'(X)$ is finite it suffices to prove that:

$$\text{for each } Z\subseteq X \text{ such that } \mathscr{E}(Z) \text{ is infinite, } (\mathscr{A}(Z))(X) \text{ is finite.} \tag{3}$$

Note that $\mathscr{A}(Z)$ is an enforcing set and so $(\mathscr{A}(Z))(X)$ is the set $\{(U,V)\in\mathscr{A}(Z): U=X\}$.

To prove (3) we will prove that critical sets are "ultimately growing" as defined below. Our definition will be in a more general setup. Let $\tau=Y_1,Y_2,\ldots$ be an infinite sequence of finite nonempty languages, and let $\rho=W_1,W_2,\ldots$ be an infinite sequence of finite nonempty languages such that for each $n\geqslant 1$, W_n is $(n+1)$-critical for τ. Then the pair (ρ,τ) is a *critically matching pair*, or a *cm pair* for short.

Claim 5. *Let (ρ,τ) be a cm pair with $\rho=W_1,W_2,\ldots$ Then for each positive integer r there exists a positive integer n_r such that, for each $n>n_r$, $|W_n|>r$.*

Proof of Claim 5. Let $\tau=Y_1,Y_2,\ldots$.

We prove this claim by contradiction. Assume to the contrary that:
there exists a positive integer m_0 such that, for infinitely many n,

$$|W_n|=m_0. \tag{4}$$

(i) Clearly for each infinite sequence $\alpha=i_1,i_2,\ldots$ of positive integers such that $i_1<i_2<\cdots$, the pair $(\rho_\alpha,\tau_\alpha)$, where $\rho_\alpha=W_{i_1},W_{i_2},\ldots$ and $\tau_\alpha=Y_{i_1},Y_{i_2},\ldots$, is critically matching.
(ii) Consequently (by (4)) there exists a cm pair (ρ',τ') such that, for each set W in ρ', $|W|=m_0$.
(iii) Also, since the first set in τ' (as all other sets in τ') is finite, there exists an element c of this set such that c belongs to infinitely many sets in ρ'.
(iv) Consequently, again by (i), there exists a cm pair (ρ'',τ'') such that, there exists an element c (of the first set of τ'') and a positive integer m_0, such that, for each element $W\in\rho''$, $|W|=m_0$ and $c\in W$.
(v) Let $\rho''=U_1,U_2,\ldots$ and $\tau''=X_1,X_2,\ldots$. Let then $\bar{\rho}=U_1-\{c\},U_2-\{c\},\ldots$. Then $(\bar{\rho},\tau'')$ is also a cm pair which is seen as follows:

- Since, for each $i\geqslant 1$, $U_i\cap X_{i+1}=\emptyset$, also $(U_i-\{c\})\cap X_{i+1}=\emptyset$.
- Since, for each $i\geqslant 1$, $c\in U_i$, it must be that $c\notin X_{i+1}$, for each $i\geqslant 1$. Thus, $U_i\cap X_j\neq\emptyset$ for all $j<i$ implies that $U_i-\{c\}\cap X_j\neq\emptyset$ for all $j<i$.

(vi) Thus $(\tilde{\rho}, \tau'')$ is a cm pair such that, for each set W in ρ, we have $|W| = m_0 - 1$.
(vii) Hence our assumption (4) implies that there exists a cm pair (ρ', τ') such that, for each set W in ρ', we have $|W| = m_0$. This in turn implies that there exists a cm pair $(\tilde{\rho}, \tau'')$ such that, for each set $W \in \tilde{\rho}$, we have $|W| = m_0 - 1$. Iterating this (decrease of constant set cardinality) leads to obvious contradiction.

Consequently (4) cannot hold, and so the claim holds. □

Claim 5 implies that, for each $Z \subseteq X$ such that $\mathscr{E}(Z)$ is infinite, there exists only a finite number of critical sets W for which $Z \cup W = X$. Hence (3) holds.

Thus $\mathscr{E}'(X)$ is finite, and since X is an arbitrary set relevant for $\mathscr{E}'$, $\mathscr{E}'$ is finitary. Hence Claim 4 holds. □

Then, Claims 3 and 4 imply the theorem. □

6. Forbidding–enforcing systems

We combine now forbidding and enforcing, and define forbidding–enforcing systems.

Definition 6.1. A *forbidding–enforcing system*, *fe system* for short, is a 3-tuple $\Gamma = (\Sigma, \mathscr{F}, \mathscr{E})$, where Σ is an alphabet, $\mathscr{F}$ is a forbidding set over Σ, and $\mathscr{E}$ is an enforcing set over Σ.

The language family defined by a fe system is defined as follows.

Definition 6.2. Let $\Gamma = (\Sigma, \mathscr{F}, \mathscr{E})$ be a fe system. A language $K \subseteq \Sigma^*$ *obeys* Γ, written K *obs* Γ, iff K *con* $\mathscr{F}$ and K *sat* $\mathscr{E}$. Then $\mathscr{L}(\Gamma)$ denotes the family of all languages obeying Γ—it is referred to as a *fe family*.

If K does not obey Γ, then we write K *nobs* Γ.

The notion of a finitary fe system will play an important role in the sequel of this paper.

Definition 6.3. A fe system $\Gamma = (\Sigma, \mathscr{F}, \mathscr{E})$ is *finitary* if $\mathscr{E}$ is finitary.

7. Satisfiability problem

We will consider now the satisfiability problem for Boolean formulas in 3-Conjunctive Normal Form (3CNF), see, e.g. [4].

Let $X = \{x_1, \ldots, x_n\}$ be the set of variables. We code each variable $x_m, 1 \leqslant m \leqslant n$, by the word Vm' where V is a distinguished letter, and m' is the representation of m in the decimal notation such that the length of the representation is the same for all variables (in this way the representation of one variable is never a subword of the representation

of another variable). Such equal length representation m' can be obtained by, e.g., adding (whenever necessary) "padding" zeros preceding the "concise" representation of m in decimal notation. Let $Z=\{z_1,z_2,\ldots,z_n\}$ be the set of codes for all variables from X, where z_m codes x_m.

Using this coding we represent now the assignment of a value (0 or 1) to a variable x_m by setting this value in front of the coding Vm' of x_m. Thus assigning 0 to x_m is represented by $0Vm'$ and assigning 1 to x_m is represented by $1Vm'$. Then an assignment ψ of Boolean values to X is represented by the set $\Psi=\{i_1V1',i_2V2',\ldots,i_nVn'\}$ where for each $1\leqslant m\leqslant n$, $\psi(x_m)=s$ iff $sVm'\in\Psi$. Let $L_X=\{1z:z\in Z\}\cup\{0z:z\in Z\}$—hence L_X consists of all representations of all possible values assigned to variables from X.

Now let $\mathscr{F}_X=\{\{1z,0z\}):z\in Z\}$ and $\mathscr{E}_X=\{(\emptyset,\{1z,0z\}):z\in Z\}$. Thus the forbidding set $\mathscr{F}_X$ does not allow any language $K\in\mathscr{L}(\mathscr{F}_X)$ to have *both* $1z$ and $0z$ in $sub(K)$ (our interpretation: one cannot assign both 1 and 0 to the same variable!). On the other hand, the enforcing set $\mathscr{E}_X$ forces any $K\in\mathscr{L}(\mathscr{E}_X)$ to have at *least one* of element of $\{1z,0z\}$ to be present in K for each $z\in Z$ (our interpretation: one has to assign either 1 or 0 to each variable!).

Hence the fe system $\Gamma_X=(L_X,\mathscr{F}_X,\mathscr{E}_X)$ is such that each $K\in\mathscr{L}(\Gamma_X)$ contains exactly one (representation of the) value assigned to each variable from X.

Note that Γ_X depends only on X (and the representation used).

We move now to represent Boolean formulas in 3CNF. Let Φ be such a formula with k clauses: $\Phi=C_1\wedge\cdots\wedge C_k$, where each clause C is of the form $\ell_{i_1}\vee\ell_{i_2}\vee\ell_{i_3}$ with $1\leqslant i_1,i_2,i_3\leqslant n$, and $i_r\neq i_s$ for $r\neq s$, $r,s\in\{i_1,i_2,i_3\}$, where for each $1\leqslant j\leqslant 3$, ℓ_{i_j} is a literal, $\ell_{i_j}\in\{x_{i_j},\neg x_{i_j}\}$. We code each such clause C by the forbidder $F(C)=\{t_{i_1}z_{i_1},t_{i_2}z_{i_2},t_{i_3}z_{i_3}\}$, where for each $1\leqslant j\leqslant 3$, $t_{i_j}\in\{0,1\}$ and $t_{i_j}=1$ iff $\ell_{i_j}=\neg x_{i_j}$. Then the forbidding set $\mathscr{F}_\Phi=\{F(C):C$ is a clause of $\Phi\}$. Finally, the fe system for Φ is $\Gamma_\Phi=(L_X,\mathscr{F}_X\cup\mathscr{F}_\Phi,\mathscr{E}_X)$.

To illustrate the above, consider the set $X=\{x_1,x_2,x_3,x_4,x_5\}$. Then we set $Z=\{V1,V2,V3,V4,V5\}$ and so
$\mathscr{F}_X=\{\{1V1,0V1\},\{1V2,0V2\},\{1V3,0V3\},\{1V4,0V4\},\{1V5,0V5\}\}$, and
$\mathscr{E}_X=\{(\emptyset,\{1V1,0V1\}),(\emptyset,\{1V2,0V2\}),(\emptyset,\{1V3,0V3\}),(\emptyset,\{1V4,0V4\}),$
$(\emptyset,\{1V5,0V5\})\}$.

Consider now the Boolean formula $\Phi=C_1\wedge C_2$, where $C_1=(\neg x_1\vee x_3\vee x_5)$ and $C_2=(\neg x_1\vee x_2\vee x_5)$.

We have then $F(C_1)=\{1V1,0V3,0V5\}$, $F(C_2)=\{1V1,0V2,0V5\}$, $\mathscr{F}_\Phi=\{F(C_1),F(C_2)\}$, and finally $\Gamma_\Phi=(L_X,\mathscr{F}_X\cup\mathscr{F}_\Phi,\mathscr{E}_X)$.

The forbidding–enforcing system Γ_Φ is "semantically" related to Φ by the following property: Φ is satisfiable iff $\mathscr{L}(\Gamma_\Phi)$ is not empty.

This is seen as follows.

Assume that Φ is satisfiable.

Let ψ be an assignment, $\psi=(x_1=i_1,\ldots,x_n=i_n)$ that satisfies Φ. Then let $\Psi=\{i_1z_1,i_2z_2,\ldots,i_nz_n\}$ (recall that $z_1,\ldots,z_n$ are the codes for variables $x_1,\ldots,x_n$, respectively).

Clearly $\Psi\ con\ \mathscr{F}_X$ and $\Psi\ sat\ \mathscr{E}_X$.

Since ψ satisfies Φ, ψ satisfies each clause of Φ, and consequently $\Psi\ con\ \mathscr{F}_\Phi$ (each forbidder of $\mathscr{F}_\Phi$ guards over one clause: it forbids the assignments that do not satisfy this clause).

Thus $\Psi\ obs\ \Gamma_\Phi$, and so $\mathscr{L}(\Gamma_\Phi) \neq \emptyset$.

Assume that $\mathscr{L}(\Gamma_\Phi)$ is not empty.

Let $K \in \mathscr{L}(\Gamma_\Phi)$, and consider $W = K \cap L_X$.

Since $K\ con\ \mathscr{F}_X$ and $K\ sat\ \mathscr{E}_X$, W contains representation of exactly one value for each variable from X, say $W = \{i_1 z_1, \ldots, i_n z_n\}$ for some $i_1, \ldots, i_n \in \{0, 1\}$.

Since $K\ con\ \mathscr{F}_\Phi$, the assignment $x_1 = i_1, \ldots, x_n = i_n$ must satisfy Φ. Thus Φ is satisfiable.

8. DNA molecules

Single DNA strands are polymers which are built from "simple" monomers, viz., nucleotides (see, e.g., [2, 6]). There are four types of nucleotides, called A, C, G, and T. Nucleotides can form strands (chains) by establishing strong (phosphodiester) bonds between "consecutive" elements.

These strands have a polarity (orientation) in the sense that one end of a strand can be biochemically distinguished from the other one; one of them is called "5′-end" and the other "3′-end". Thus we can write down a DNA strand as a sequence over four letters (A, C, G, T) indicating which end is the 5′-end and which is the 3′-end.

For example, 5′-*AATCTGC*-3′ is a DNA strand consisting of 7 nucleotides which when read from the 5′-end towards the 3′-end forms the word (string) *AATCTGC*.

Note that in the above we can skip the indication of the 3′-end ("-3′"), because if the other end is the 5′-end, then this end must be the 3′-end. As a matter of fact, to make the notation even simpler, one often omits the explicit indication of the 5′- and 3′-ends: the convention is that the left end is the 5′-end and the right end is the 3′-end. Thus the above DNA strand is written as *AATCTGC*.

Two single DNA strands s_1, s_2 can "stick together" to form a "perfect" *double strand* if the translation of s_1 from the 5′-end towards 3′-end (nucleotide-by-nucleotide) using the translation rules $A \to T$, $C \to G$, $G \to C$, $T \to A$ yields the s_2 strand read from the 3′-end towards 5′-end. Then the first nucleotide from the 5′-end of s_1 will stick to the first nucleotide from the 3′-end of s_2, the second nucleotide from the 5′-end of s_1 will stick to the second nucleotide from the 3′-end of s_2, and so on.

Thus the strand $s_2 =$ 5′-*AATCTGC* from above will stick together with the strand $s_2 =$ 3′-*TTAGACG*-5′ forming the double stranded molecule given in Fig. 1.

5′-*AATCTGC*-3′
3′-*TTAGACG*-5′

Fig. 1.

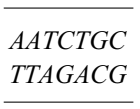

Fig. 2.

Fig. 3.

Again, to make the notation simpler we may omit the explicit indication of the 5′- en 3′-ends: the convention is that the "upper" strand is written in the 5′-3′ orientation and the "lower" strand is (thus) written in the 3′-5′ orientation; we refer to this representation as the *standard representation*. Thus the above double strand may be written as in Fig. 2.

The bonds between complementary nucleotides (A with T, and C with G) on the "opposite" strands are weak (hydrogen) bonds.

To summarize: such a double stranded DNA molecule (duplex) is formed by strong phosphodiester bonds within each single strand and weak hydrogen bonds between the two strands. The hydrogen bonds are formed always between complementary nucleotides, where the complementary pairs are $\{A, T\}$ and $\{C, G\}$.

To avoid too many technical details we will first consider only double stranded molecules.

Double strands may be also "incomplete", as e.g., the one in Fig. 3.

It has an "overhanging" 3′-end, viz. 5′-*TGC*. Such an end is also called a "sticky end" because if the single strand 3′-*ACG* is available, then it may stick to (anneal to) the sticky end *TGC* forming the complete double stranded molecule given in Fig. 2.

In reality this molecule will not be complete, because 3′-*ACG* will stick to the sticky end *TGC only* by the (weak) hydrogen bonding, while the phosphodiester bond between *G* (at the 3′-end of 5′-*TTAG* in the lower strand) and *A* (at the 3′-end of 3′-*ACG* connected to the sticky end) is missing. This is resolved by having in the solution containing the molecule the enzyme called ligase that (eagerly!) repairs such "nicks" by establishing the missing phosphodiester bonds.

A double strand may be also incomplete because some of the nucleotides in one strand, but not at one of the ends of the strand, may be missing their complementary partners—this result in *gaps*. For example, the molecule represented by the double string in Fig. 4 has three gaps: one in one strand and two in the other one; it has also a sticky 5′-end *TTA*.

Let us consider once again the representation of double stranded DNA molecules by double strings, as, e.g., the one in Fig. 2. Clearly, if we take the mirror image of this representation and then exchange upper and lower strands, then we again get a

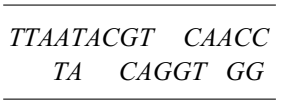

Fig. 4.

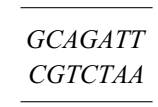

Fig. 5.

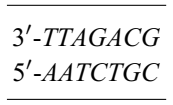

Fig. 6.

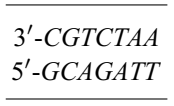

Fig. 7.

representation of the same molecule. We obtain in this way the representation from Fig. 5.

This is another representation of the same molecule. Since we are interested in the molecules, and there is nothing that distinguishes one representation from the other, *both representations must be present in the representation of a* "*molecular soup*" *containing this molecule*.

More formally, given a representation w_1 of a molecule m, one obtains the other representation w_2 of m by the composition of two operations: the *mirror image*, *mir*, and the *exchange*, *ex*. The mirror image of w_1 yields w_1 read from right to left, and the exchange, as the name suggests, exchanges the upper string with the lower string. Thus if w_1 is the representation from Fig. 2, then $ex(w_1)$ is the double string from Fig. 6, while $mir(w_1)$ is the double string from Fig. 7.

Now, $mir(ex(w_1))$ is the double string w_2 from Fig. 5.

Clearly, $ex(mir(w_1))$ is the same double string.

Now, the composition of exchange and mirror image operations (in any order because the result is the same) is the *inversion* operation, denoted by *inv*. Thus in our example, $inv(w_1) = mir(ex(w_1)) = ex(mir(w_1)) = w_2$.

In general, $inv(w)$ does not have to be different from w. If $inv(w) = w$, then the double string w is a *palindrome* (and the molecule m represented by w is also called a *palindrome*).

Note that, if w is a representation of a double stranded molecule, then neither $mir(w)$ nor $ex(w)$ is a representation (of any molecule) because in both of them the upper strand has the 3′–5′ orientation rather than 5′–3′ orientation as required in the standard representation. For this reason in Figs. 6 and 7 we have explicitly indicated the 3′-end and the 5′-end for the upper and the lower string, respectively.

We proceed now to represent double stranded (perhaps incomplete) molecules in a more systematic way.

First we establish the alphabet we need: let $\Delta = \{A, C, G, T\}$.

Let $\Theta_d = \{\tilde{X} : X \in \Delta\}$, where $\tilde{A}, \tilde{C}, \tilde{G}, \tilde{T}$ are the following double strand letters:

$$\tilde{A} = \begin{pmatrix} A \\ T \end{pmatrix}, \quad \tilde{C} = \begin{pmatrix} C \\ G \end{pmatrix}, \quad \tilde{G} = \begin{pmatrix} G \\ C \end{pmatrix}, \quad \text{and} \quad \tilde{T} = \begin{pmatrix} T \\ A \end{pmatrix}.$$

Each of these letters represents a letter in the upper row and "sticking to it" the complementary letter in the lower row.

Let $\Theta_u = \{\hat{X} : X \in \Delta\}$.

Each of these ("upper") letters represents a letter in the upper row without a complementary letter sticking to it in the lower row.

Let $\Theta_\ell = \{\check{X} : X \in \Delta\}$.

Each of these ("lower") letters represents a letter in the lower row without a complementary letter sticking to it in the upper row.

Finally, let $\Theta = \Theta_d \cup \Theta_u \cup \Theta_\ell$.

If a catenation of letters from Θ corresponds to the standard representation of a molecule, then the translation into the standard representation is obvious. Thus, e.g., $\tilde{A}\tilde{A}\tilde{T}\tilde{C}\tilde{T}\tilde{G}\tilde{C}$ translates into the representation from Fig. 2, $\tilde{A}\tilde{A}\tilde{T}\tilde{C}\hat{T}\hat{G}\hat{C}$ translates into the representation from Fig. 3, and $\hat{T}\hat{T}\hat{A}\tilde{A}\tilde{T}\hat{A}\hat{C}\tilde{G}\tilde{T}\check{G}\tilde{C}\tilde{A}\hat{A}\tilde{C}\tilde{C}$ translates into the representation from Fig. 4. However not all catenations of letters from Θ correspond to the standard representation. The reason is that if a word in Θ^+ contains a catenation pq where either $p \in \Theta_u$ and $q \in \Theta_\ell$, or $p \in \Theta_\ell$ and $q \in \Theta_u$, then nucleotides corresponding to these consecutive letters would not be bonded by either phosphodiester or hydrogen bonds!!

Having this in mind we may consider now also single stranded molecules. We assume that for a word $w = X_1 \ldots X_n$, with $n \geqslant 1$ and $X_1, \ldots, X_n \in \Delta$, the word $\hat{w} = \hat{X}_1 \ldots \hat{X}_n$ represents the molecule 5′-$X_1 \ldots X_n$-3′, and the word $\check{w} = \check{X}_1 \ldots \check{X}_n$ represents the molecule 3′-$X_1 \ldots X_n$-5′. Note that we use here "cups" $\check{}$ and "hats" $\hat{}$ over the letters from Δ to represent directionality.

Now the inversion operation can be formally defined for *all* words in Θ^+ as follows.

Let γ be the following (*complementarity*) homomorphism defined on Θ by:
$\gamma(\tilde{A}) = \tilde{T}$, $\gamma(\tilde{T}) = \tilde{A}$, $\gamma(\tilde{C}) = \tilde{G}$, $\gamma(\tilde{G}) = \tilde{C}$,
$\gamma(\hat{X}) = \check{X}$ for all $X \in \Delta$, and
$\gamma(\check{X}) = \hat{X}$ for all $X \in \Delta$.

Then for each $w \in \Theta^+$, $inv(w)$ is defined by: $inv(w) = mir(\gamma(w))$.

Note that through this definition of γ, also single strands have two representations: one written (from left-to-right) in the 5′–3′ direction—this is over the alphabet

Θ_u, and one written (also from left-to-right) in the $3'$–$5'$ direction—this is over the alphabet Θ_ℓ.

We are ready now to translate all of the above into the formal framework of forbidding–enforcing systems.

We assume that Θ is the alphabet of the fe system Γ that we consider to describe the structure of DNA molecules and some operations on them. However the definition of Γ can be easily modified if Θ would be only a subset of the alphabet Σ of Γ (when Γ would be describing a "bigger" system which among others deals with DNA molecules and the operations on them that we consider here).

We begin by describing the structure of DNA molecules.

The forbidding set $\mathscr{F}_\Theta$ is defined by

$\mathscr{F}_\Theta = \{\{\hat{X}\check{Y}\} : X, Y \in \Delta\} \cup \{\{\check{X}\hat{Y}\} : X, Y \in \Delta\}$.

We also need the enforcing set $\mathscr{E}_\Theta$ defined by

$\mathscr{E}_\Theta = \{(\{w\}, \{inv(w)\}) : w \in \Theta^+\}$.

Hence the forbidding set $\mathscr{F}_\Theta$ forbids the catenation of letters from Θ_u with letters from Θ_d in any order.

The enforcing set $\mathscr{E}_\Theta$ takes care of the fact that each molecule may be represented in two ways in Θ^+.

Formally this is expressed as follows (where *id* is the identity function).

Property 1. Let $E = (\{u_1, \ldots, u_r\}, \{v_1, \ldots, v_s\})$ be an enforcer, where $u_1, \ldots, u_r$, $v_1, \ldots, v_s \in \Theta^+$, and let the *closure of* E be the set

$clos(E) = \{(\{\varphi_1(u_1), \ldots, \varphi_r(u_r)\}, \{\psi_1(v_1), \ldots, \psi_s(v_s)\})$: for each $1 \leqslant i \leqslant r$ and each $1 \leqslant j \leqslant s$, $\varphi_i, \psi_j \in \{inv, id\}\}$.

For every language $K \subseteq \Theta^+$, if K *sat* $(\mathscr{E}_\Theta \cup \{E\})$, then K *sat* $(\mathscr{E}_\Theta \cup \{E'\})$ for each $E' \in clos(E)$.

Proof. Let $K \subseteq \Theta^+$, let $\varphi_1, \ldots, \varphi_r, \psi_1, \ldots, \psi_s \in \{inv, id\}$ and let $E' = (\{\varphi_1(u_1), \ldots, \varphi_r(u_r)\}, \{\psi_1(v_1), \ldots, \psi_s(v_s)\})$.

Assume that $\{\varphi_1(u_1), \ldots, \varphi_r(u_r)\} \subseteq K$; otherwise E' is trivially satisfied by K.

Since K *sat* $\mathscr{E}_\Theta$, $u_1, \ldots, u_r \in K$.

Since K *sat* E, $v_j \in K$ for some $j \in \{1, \ldots, s\}$.

Since K *sat* $\mathscr{E}_\Theta$, $\psi_j(v_j) \in K$, and so K *sat* E'.

Thus K *sat* $(\mathscr{E}_\Theta \cup \{E'\})$. □

Property 2. Let $F = \{u_1, \ldots, u_r\}$ be an arbitrary forbidder, where $u_1, \ldots, u_r \in \Theta^+$, and let the closure of F be the set $clos(F) = \{\{\varphi_1(u_1), \ldots, \varphi_r(u_r)\}$: for each $1 \leqslant i \leqslant r$, $\varphi_i \in \{inv, id\}\}$. For every language $K \subseteq \Theta^+$, if K *sat* $\mathscr{E}_\Theta$ and K *con* F, then K *con* F' for each $F' \in clos(F)$.

Proof. Let $K \subseteq \Theta^+$, let $\varphi_1, \ldots, \varphi_r \in \{inv, id\}$ and let $F' = \{\varphi_1(u_1), \ldots, \varphi_r(u_r)\}$.

Since K *con* F, $u_i \notin sub(K)$ for some $i \in \{1, \ldots, n\}$.

Since K *sat* $\mathscr{E}_\Theta$, $\varphi_i(u_i) \notin sub(K)$ and so K *con* F'. □

Fig. 8.

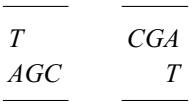

Fig. 9.

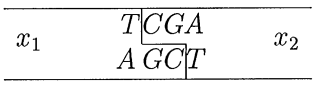

Fig. 10.

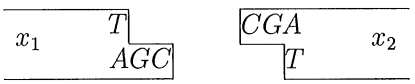

Fig. 11.

Restriction enzymes are enzymes that are very useful in genetic engineering and in particular in DNA computing. The restriction enzymes that we are interested in work as follows.

A restriction enzyme seeks a particular combination of nucleotides on a double stranded DNA molecule—this is called the *restriction site* for the enzyme (for the purpose of this paper we may assume that such a site is *unique* for a given restriction enzyme). After finding its restriction site, the enzyme attaches to it and cuts the double strand within it. Such a cut can be either straight (blunt) or staggered. Thus, e.g., for the enzyme Tag1 (see, e.g., [6]) the restriction site is given in Fig. 8.

The cut by Tag1 is staggered: the restriction site is cut as in Fig. 9.

Hence a molecule of the form given in Fig. 10 will be cut into two molecules given in Fig. 11.

The effect of Tag1 is formally described by the enforcing set

$$\mathscr{E}_{Tag1} = \{(\{w_1\tilde{T}\tilde{C}\tilde{G}\tilde{A}w_2\}, \{w_1\tilde{T}\check{G}\check{C}\}) : w_1, w_2 \in \Theta^*\}.$$

Note that $\mathscr{E}_{Tag1}$ "enforces" one molecule resulting from the cut by Tag1 (the left one in Fig. 11), but $\mathscr{E}_{\Theta}$ ensures that the other one must be also in.

As we have mentioned already, ligase is an enzyme that repairs nicks by establishing the missing phosphodiester bonds. Thus, if, e.g., the two molecules from Fig. 11 will get (very) close to each other, then the hydrogen bonds will position the sticky end GC of the "left" molecule below the sticky end CG of the "right" molecule, and then

the ligase present in the solution will seal the two nicks producing one molecule, as in Fig. 10.

This ligating in general (for arbitrary molecules with the sticky ends "fitting" into the restriction site of Tag1) is formally described by the enforcing set $\mathscr{E}_{(ligase,Tag1)}$ defined as follows:

$$\mathscr{E}_{(ligase,Tag1)} = \{(\{w_1\tilde{T}\check{G}\check{C}, \hat{C}\hat{G}\tilde{A}w_2\}, \{w_1\tilde{T}\tilde{C}\tilde{G}\tilde{A}w_2\}) : w_1, w_2 \in \Theta^*\}.$$

9. Computation trees for forbidding–enforcing systems

A fe system is given by its forbidding and enforcing sets—each of them may be of arbitrary cardinality, it may be also infinite. Analyzing fe systems is thus a highly combinatorial task—one has to evaluate the effect of all enforcers which have to be applied in such a way that they do not violate constraints set up by forbidders. Therefore an important research task is to look for a structure behind all the computations (evolutions) in a fe system.

Indeed, it turns out that all computations and their effects (languages) in a finitary fe system can be elegantly represented by trees.

We associate a tree with a fe system as follows (in this section, by a *path* (in a graph) we mean a rooted path).

Definition 9.1. Let $\Gamma = (\Sigma, \mathscr{F}, \mathscr{E})$ be a fe system, and let τ be a tree. Then τ is a *Γ-tree* iff

(1) Each node label is a finite language over Σ,
(2) If $X \subseteq \Sigma^+$ is a node label, then $X\ con\ \mathscr{F}$,
(3) If nodes v_1, v_2 are on the same path with $v_1 <_\tau v_2$, then $lab_\tau(v_1) \subseteq lab_\tau(v_2)$ and $lab_\tau(v_1) \vdash_{\mathscr{E}} lab_\tau(v_2)$.

Definition 9.1 is illustrated in Fig. 12, where K_1, K_2 are finite, $K_1\ con\ \mathscr{F}$, $K_2\ con\ \mathscr{F}$, $K_1 \subseteq K_2$ and $K_1 \vdash_{\mathscr{E}} K_2$.

Complete Γ-trees are of special interest to us because they represent all languages from $\mathscr{L}(\Gamma)$.

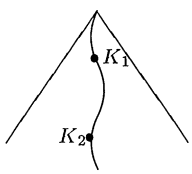

Fig. 12.

Definition 9.2. Let $\Gamma=(\Sigma,\mathscr{F},\mathscr{E})$ be a fe system. A Γ-tree τ is a *complete* Γ-tree iff, for every $K\in\mathscr{L}(\Gamma)$,
(1) If K is finite, then K is a node label in τ.
(2) If K is infinite, then there exists a complete path π in τ such that $K=\bigcup_{v\in\pi} lab_\tau(v)$.

Here is our representation theorem for finitary fe systems.

In order to somewhat simplify the constructions in the proof of the representation theorem, we will consider within this proof the *inclusive version* of the $\mathscr{E}$-extension relation, i.e., whenever we write $K_1\vdash_{\mathscr{E}} K_2$, we also assume that $K_1\subseteq K_2$ (this carries also to minimal $\mathscr{E}$-extensions). It is really easy, but somewhat tedious, to modify the proof for the case when the inclusive version assumption is not made.

Note that in a Γ-tree τ if $v_1<_\tau v_2$, then $lab_\tau(v_1)\subseteq lab_\tau(v_2)$, and so when we consider Γ-trees, then we consider de facto the inclusive version of the extension relation.

Theorem 9.1. *For each finitary fe system Γ there exists a complete Γ-tree.*

Proof. Let $\Gamma=(\Sigma,\mathscr{F},\mathscr{E})$ be a finitary fe system. We will prove the theorem by first giving a construction of a node labeled tree τ, and then proving that τ is a complete Γ-tree.

CONSTRUCTION 1. Let $\tau=(V,E,\varphi)$ be the node labeled tree, with φ labeling nodes by finite languages, constructed as follows.
(1) Let r be the root of τ. Then $\varphi(r)=\emptyset$, and $lev(r)=0$ (lev is the *level* function defined on V inductively throughout our construction).
(2) Let $v\in V$ with $\varphi(v)=A$ and $lev(v)=n$ for some $n\geqslant 0$. We consider separately two cases.
(2.1) *A nsat $\mathscr{E}$.*

Let $\mathscr{U}=\{U: U \text{ is } (A,\vdash_{\mathscr{E}})\text{-minimal}\}$. Clearly, for each $U\in\mathscr{U}$, $U\subseteq\bigcup_{Z\subseteq A}\bar{\mathscr{E}}(Z)$. Since $\mathscr{E}$ is finitary, and A is finite, this implies that $\mathscr{U}$ is finite; let $\mathscr{U}=\{U_1,\ldots,U_m\}$ for some $m\geqslant 1$.

Let $M_1=\{U: U \text{ con } \mathscr{F} \text{ and } U\in\mathscr{U}\}$, $M_2=\{U: U \text{ con } \mathscr{F} \text{ and } U=U'\cup\{w\}$ for some $U'\in\mathscr{U}$ and $w\in\Sigma^+$ with $|w|\leqslant n+1\}$, and
let $M=M_1\cup M_2$, say $M=\{B_1,\ldots,B_k\}$ for some $k\geqslant 1$. We set then $ddes_\tau(v)=\{v_1,\ldots,v_k\}$, with $\varphi(v_i)=A\cup B_i$ and $lev(v_i)=n+1$, for each $1\leqslant i\leqslant k$ (note that, because we consider the inclusive version of the $\mathscr{E}$-extension, $A\subseteq\varphi(v_i)$ for all $1\leqslant i\leqslant k$).
(2.2) *A sat $\mathscr{E}$.*

Let $M_1=\{A\}$,

$M_2=\{B: B \text{ con } \mathscr{F}$ and $B=A\cup\{w\}$ for some $w\in\Sigma^+$ with $|w|\leqslant n+1\}$, and let $M=M_1\cup M_2$. Clearly M is a finite set; let $M=\{B_1,\ldots,B_k\}$ for some $k\geqslant 1$. We set now $ddes_\tau(v)=\{v_1,\ldots,v_k\}$ with $\varphi(v_i)=B_i$ and $lev(v_i)=n+1$, for each $1\leqslant i\leqslant k$.

The intuition behind the construction of τ is as follows. If A does not satisfy $\mathscr{E}$, then we consider the set $U_1,\ldots,U_m$ of all minimal $\mathscr{E}$-extensions of A. Then, for each

U_i, $1 \leqslant i \leqslant m$, we create a direct descendant of v with the label U_i provided that U_i is consistent with $\mathscr{F}$; note that since we consider the inclusive version of the $\mathscr{E}$-extension relation, $A \subseteq U_i$ for each $1 \leqslant i \leqslant m$. Moreover we create a direct descendant of v for each set U which is consistent with $\mathscr{F}$ and is of the form $U_i \cup \{w\}$, where $1 \leqslant i \leqslant m$ and $|w| \leqslant n+1$, where $n = lev(v)$.

If A satisfies $\mathscr{E}$, then A is already its own minimal extension, and so we create direct descendants of v *only* for the sets of the form $A \cup \{w\}$, where $|w| \leqslant n+1$ for $n = lev(v)$, and $A \cup \{w\}$ is consistent with $\mathscr{F}$, and for A itself—thus we create a copy of A. This copy is needed because if, e.g., $A \cup \{w\}$ is consistent with $\mathscr{F}$, where $|w| = 100$, then we have to have a copy of A available on the level 99 in order to create a node labelled by $A \cup \{w\}$. Since in a Γ-tree each node has only a finite number of direct descendants we cannot add "at once" to A all words of length not bounded in advance! Hence carrying over a copy of A is handy and necessary.

Adding in advance (in both cases) words of length bounded by the level of a considered node, allows one "eventually" to create all "good" finite extensions of all sets that label the nodes of τ.

We will prove now through a sequence of lemmas that τ is a complete Γ-tree. □

Lemma 9.1. *τ is a Γ-tree.*

Proof. (1) It follows directly from the construction that, for each $v \in V$, $\varphi(v)$ is a finite language.

(2) It also follows easily from the construction (by induction on the level of a node) that, for each $v \in V$, $\varphi(v)\, con\, \mathscr{F}$.

(3) Assume now that u_1, u_2 are two nodes of τ such that $u_2 \in ddes(u_1)$. By CONSTRUCTION 1 it follows immediately that $\varphi(u_1) \subseteq \varphi(u_2)$. Also

- if Case (2.1) (from CONSTRUCTION 1) holds, then (by the definition of $\mathscr{U}$) it follows that $\varphi(u_1) \vdash_{\mathscr{E}} \varphi(u_2)$,
- if Case (2.2) holds, then (because $A \vdash_{\mathscr{E}} A$), it also follows that $\varphi(u_1) \vdash_{\mathscr{E}} \varphi(u_2)$.

Since $\subseteq$ and $\vdash_{\mathscr{E}}$ are transitive relations (remember that we consider the inclusive version of $\vdash_{\mathscr{E}}$), it follows from the above that if $u_1, u_2 \in V$ are such that $u_1 <_\tau u_2$, then $\varphi(u_1) \subseteq \varphi(u_2)$ and $\varphi(u_1) \vdash_{\mathscr{E}} \varphi(u_2)$.

Now it follows from (1)–(3) that τ is a Γ-tree, and so Lemma 9.1 holds. □

We need the following definitions of convergence ("within Γ") before we can formulate our next lemma.

Definition 9.3. Let $\sigma = K_0, K_1, \ldots$ be an infinite ascending sequence of finite languages and let K be a language. We say that σ *converges to* K iff $\bigcup \sigma = K$, and for each $n \geqslant 0$, $K_n \vdash_{\mathscr{E}} K_{n+1}$.

Note that this notion of convergence depends on $\mathscr{E}$. However we use the term "converges" rather than "$\mathscr{E}$-converges", because $\mathscr{E}$ is fixed within the proof of Theorem 9.1.

We assume that the alphabet Σ of Γ is ordered, and so all the words in Σ^* are ordered lexicographically—we use lex_Σ to denote this order.

Then for every language K over Σ we define $\min_\Sigma(K)$ as the singleton set containing the minimal w.r.t. lex_Σ element of K if $K \neq \emptyset$, and as $\emptyset$ if $K = \emptyset$.

Definition 9.4. Let $\sigma = K_0, K_1, \ldots$ be an ascending sequence of languages and let K be a language. We say that σ *slowly converges to* K iff

(i) σ converges to K,
(ii) $K_0 = \emptyset$, and
(iii) for each $n \geqslant 0$, there exists a language L such that
  (a) L is $(K_n, \vdash_{\mathscr{E}})$-minimal, and
  (b) $K_{n+1} = L \cup \min_\Sigma(\{w \in K - K_n : |w| \leqslant n+1\})$.

Lemma 9.2. *Let $K \in \mathscr{L}(\Gamma)$. There exists a sequence of languages σ which slowly converges to K.*

Proof. We will first construct $\sigma = K_0, K_1, \ldots, K_n, \ldots$ by induction on n.

CONSTRUCTION 2. (1) Set $K_0 = \emptyset$.

(2) Assume that a finite $K_n \subseteq K$ is defined.

To construct K_{n+1} we will consider separately two cases.

(2.1) K_n *nsat* $\mathscr{E}$.

Since K *sat* $\mathscr{E}$, $\mathscr{E}$ is finitary, and K_n is finite, there exists a finite language L such that $K_n \subseteq L \subseteq K$ and $K_n \vdash_{\mathscr{E}} L$ (see the proof of Theorem 4.2). Let L_0 be a minimal (w.r.t. $\subseteq$) such L, and let then $K_{n+1} = L_0 \cup \min_\Sigma(\{w \in K - K_n : |w| \leqslant n+1\})$.

(2.2) K_n *sat* $\mathscr{E}$.

We let then $K_{n+1} = K_n \cup \min_\Sigma(\{w \in K - K_n : |w| \leqslant n+1\})$.

Note that it may be that $K_{n+1} = K_n$ (if there are no words in $K - K_n$ of length not exceeding $n+1$).

We will prove now that σ slowly converges to K.

Claim 1. *σ converges to K.*

Proof of Claim 1. Clearly each K_n, $n \geqslant 0$, is a finite language, and $K_n \subseteq K_{n+1}$.

Now let us consider $\bigcup \sigma$ and assume that $\bigcup \sigma \neq K$.

From the construction of σ it follows that $\bigcup \sigma \subseteq K$, and so if $\bigcup \sigma \neq K$, then $K - \bigcup \sigma \neq \emptyset$. Let then $z = \min_\Sigma(K - \bigcup \sigma)$, and let $m = |z|$. But then, by the construction of σ, $z \in K_m$, and hence $z \in \bigcup \sigma$—a contradiction.

Thus it must be that $\bigcup \sigma = K$.

Now consider K_n and K_{n+1} for arbitrary $n \geqslant 0$.

If Case (2.1) of Construction 2 holds, then $K_n \vdash_{\mathscr{E}} K_{n+1}$, because $K_n \vdash_{\mathscr{E}} L_0$ and $L_0 \subseteq K_{n+1}$.

If Case (2.2) of Construction 2 holds, then $K_n \vdash_{\mathscr{E}} K_{n+1}$, because K_n *sat* $\mathscr{E}$ and $K_n \subseteq K_{n+1}$.

Hence Claim 1 holds.

Claim 2. *σ slowly converges to K.*

Proof of Claim 2. (i) By Claim 1, σ converges to K.

(ii) By the construction of σ, $K_0 = \emptyset$.

(iii) Let $n \geqslant 0$.

If Case (2.1) of CONSTRUCTION 2 holds, then $L = L_0$ will satisfy point (iii) of Definition 9.4. If Case (2.2) of CONSTRUCTION 2 holds, then $L = K_n$ will satisfy point (iii) of Definition 9.4.

Now the claim holds from (i), (ii), and (iii). □

CONSTRUCTION 2 and Claim 2 imply Lemma 9.2. □

Lemma 9.3. *Let $K \in \mathscr{L}(\Gamma)$. If σ is a sequence of languages slowly converging to K, then there exists a complete path π of τ such that $\sigma = \varphi(\pi)$.*

Proof. Let $\sigma = K_0, K_1, \ldots, K_n, \ldots$ We will construct, by induction on n, a path $\pi = v_0, v_1, \ldots, v_n, \ldots$ of τ satisfying the statement of the lemma.

CONSTRUCTION 3. (1) Let $v_0 = r$ (the root of τ). By CONSTRUCTION 1, $\varphi(r) = \emptyset = K_0$.

(2) Assume that, for some $n \geqslant 0$, a path $\pi_n = v_0, \ldots, v_n$ such that $\varphi(\pi_n) = \sigma_n = K_0, \ldots, K_n$ has already been constructed.

To construct $\pi_{n+1} = v_0, \ldots, v_n, v_{n+1}$ we will consider separately two cases corresponding to Cases (2.1) and (2.2) from CONSTRUCTION 1.

(2.1). *K_n nsat $\mathscr{E}$.*

If $K_{n+1} = L$, where L satisfies Definition 9.4(iii), then in terms of CONSTRUCTION 1, see point (2.1), this means that $K_{n+1} \in M_1$, and so there exists $u \in ddes_\tau(v_n)$ such that $\varphi(u) = K_{n+1}$. We set $v_{n+1} = u$.

If $K_{n+1} = L \cup \{z\}$, where $\mathscr{L}$ satisfies Definition 9.4(iii), and $z = \min_\Sigma\{w \in K - K_n : |w| \leqslant n+1\}$ (see Definition 9.4(iii.b)), then in terms of CONSTRUCTION 1, see point (2.1), this means that $K_{n+1} \in M_2$, and so there exists $u \in ddes_\tau(v_n)$ such that $\varphi(u) = K_{n+1}$. We set $v_{n+1} = u$.

Consequently, if Case (2.1) holds, then we obtain a path $\pi_{n+1} = v_0, v_1, \ldots, v_{n+1}$ such that $\varphi(\pi_{n+1}) = \sigma_{n+1} = K_0, \ldots, K_{n+1}$.

(2.2). *K_n sat $\mathscr{E}$.*

If $K_{n+1} = K_n$, then in terms of CONSTRUCTION 1, see point (2.2), this means that $M_1 = \{K_{n+1}\}$ and so there exists $u \in ddes_\tau(v_n)$ such that $\varphi(u) = K_{n+1}$. We set then $v_{n+1} = u$.

If $K_{n+1} \neq K_n$, then $K_{n+1} = K_n \cup \{z\}$, where $z = \min_\Sigma(\{w \in K - K_n : |w| \leqslant n+1\})$ (see Definition 9.4(iii.b)). Then in terms of CONSTRUCTION 1, see point (2.2), this means that $K_{n+1} \in M_2$, and so there exists $u \in ddes_\tau(v_n)$ such that $\varphi(u) = K_{n+1}$. We set then $v_{n+1} = u$.

Consequently, if Case (2) holds, then we obtain a path $\pi_{n+1} = v_0, v_1, \ldots, v_{n+1}$ such that $\varphi(\pi_{n+1}) = \sigma_{n+1} = K_0, \ldots, K_{n+1}$.

This completes the inductive step of the construction of a complete path π of τ such that $\sigma = \varphi(\pi)$.

Hence Lemma 9.3 holds. □

We conclude now the proof of Theorem 9.1 by demonstrating that τ is a complete Γ-tree.

First of all, by Lemma 9.1, τ is a Γ-tree. To prove that τ is a complete Γ-tree, let us consider a language $K \in \mathscr{L}(\Gamma)$.

(i) Assume that K is finite.

By Lemma 9.2, there exists an ascending sequence of languages $\sigma = K_0, K_1, \ldots$ which slowly converges to K. Since K is finite, there exists n_0 such that $K_n = K$ for all $n \geqslant n_0$. Hence by Lemma 9.3, there exists a complete path $\pi = v_0, v_1, \ldots$ in τ such that $\varphi(v_n) = K$ for all $n \geqslant n_0$. Hence indeed K is a node label of τ, e.g., $lab_\tau(v_{n_0}) = K$.

(ii) Assume that K is infinite.

By Lemma 9.2, there exists an ascending sequence of languages σ which slowly converges to K. Hence, by Lemma 9.3, there exists a complete infinite path π such that $\sigma = \varphi(\pi)$, and so $K = \bigcup \sigma = \bigcup \varphi(\pi)$.

Now, (i) and (ii) complete the proof of Theorem 9.1. □

Theorem 9.1 does not hold for arbitrary fe systems—the restriction to finitary fe systems is very essential. Consider for example the fe system $\Gamma = (\{a, b\}, \emptyset, \mathscr{E})$, where $\mathscr{E}$ is the enforcing set from Example 2.3: $\mathscr{E} = \{(\emptyset, \{b^n\}) : n \text{ is even}\}$. Clearly, $\mathscr{E}$ is not even weakly finitary: for every pair of languages K_1, K_2, if $K_1 \vdash_{\mathscr{E}} K_2$ then K_2 is infinite. Since nodes of a Γ-tree are labelled by finite languages only, one cannot construct even a path of length two!! Thus there is no Γ-tree.

Theorem 9.1 supports our view that one should not consider arbitrary fe systems, but rather restrict oneself to finitary fe systems, where computations "happen gradually". They evolve rather than in one step deliver a whole infinite language!!

An additional attraction of finitary fe systems is that they can specify all families of fe languages (Theorem 5.1). Therefore we have the following result.

Corollary 9.1. *Let $\mathscr{K}$ be a fe family. There exists a finitely branching tree τ with nodes labelled by finite languages such that a language $K \in \mathscr{K}$ iff there exists an infinite complete path π of τ such that $K = \bigcup lab_\tau(\pi)$.*

Proof. Let Γ' be a fe system such that $\mathscr{L}(\Gamma') = \mathscr{K}$, and let $\Gamma = (\mathscr{F}, \mathscr{E})$ be an equivalent finitary fe system—see Theorem 5.1. Let then $\tau = (V, E, \varphi)$ be the complete Γ-tree constructed as in the proof of Theorem 9.1 (see CONSTRUCTION 1). We will prove now that this τ satisfies the statement of the corollary.

Let $K \in \mathscr{K}$.

Since $\mathscr{K} = \mathscr{L}(\Gamma)$ and Γ is finitary, by Lemma 9.2 there exists a sequence of languages σ which slowly converges to K. Hence by Lemma 9.3 there exists an infinite complete path π of τ such that $\sigma = \varphi(\pi)$. Since $\bigcup \sigma = K$, we get indeed $K = \bigcup \varphi(\pi)$.

Now let π be an infinite path of τ, $\pi = v_0, v_1, \ldots$, and let $K = \bigcup \varphi(\pi)$.

(i) K *con* $\mathscr{F}$.

This is seen as follows.

By CONSTRUCTION 1 of τ, for each node $v \in V$, $\varphi(v)$ *con* $\mathscr{F}$. Also by CONSTRUCTION 1, $\varphi(\pi)$ is an ascending sequence of languages. Hence, by Theorem 1.1, $K = \bigcup \varphi(\pi)$ *con* $\mathscr{F}$.

(ii) K *sat* $\mathscr{E}$.

This is seen as follows.

Let $(X, Y) \in \mathscr{E}$ be K-applicable. Hence $X \subseteq \bigcup \varphi(\pi)$, and since X is finite, $X \subseteq \varphi(v_m)$ for some $m \geqslant 0$. Since τ is a Γ-tree, $\varphi(v_m) \vdash_{\mathscr{E}} \varphi(v_{m+1})$ and so $Y \cap \varphi(v_{m+1}) \neq \emptyset$. Hence K *sat* (X, Y), and since (X, Y) is an arbitrary K-applicable enforcer from $\mathscr{E}$, K *sat* $\mathscr{E}$.

Now, (i) and (ii) imply that $K \in \mathscr{L}(\Gamma) = \mathscr{K}$. Thus Corollary 9.1. holds. □

Note that, if in the above Γ' is not finitary, then the complete Γ-tree τ will represent *all* languages of Γ' (because $\mathscr{L}(\Gamma') = \mathscr{L}(\Gamma) = \mathscr{K}$), *but* τ will represent the evolving relation $\vdash_{\mathscr{E}}$ of Γ *and not of* Γ'.

10. Discussion

In this paper we have presented some basic notions of the theory of fe systems. We believe that the forbidding–enforcing paradigm is interesting and novel for both DNA computing and formal language theory. As a matter of fact, it is an example of an interesting cross-fertilization between the two areas. We consider this paper to be merely a beginning of a systematic investigation of the forbidding–enforcing paradigm. Such an investigation should study this paradigm both within the formal language theory and within DNA computing.

(1) The theory of fe systems is an approach to a "more tolerant" language theory, where a specification merely gives forbidding and enforcing conditions and every language that obeys them fulfills the specification. One needs to investigate basic aspects of such specifications such as (other) normal forms, closure properties, the existence of generators for such families, decision problems and complexity issues. Also possible relationships between the enforcing and the forbidding sets of one fe system should be investigated.

(2) To test the suitability of fe systems for DNA computing one should attempt to use fe systems for the *description* and *analysis* of various laboratory experiments on DNA computing.

We are currently working on some of the above listed issues.

Acknowledgements

The authors are indebted to Hendrik Jan Hoogeboom, the anonymous referee, and especially to Nikè van Vugt for a very careful reading of the previous version of the manuscript and very useful comments.

References

[1] L.M. Adleman, Molecular computation of solutions to combinatorial problems, Science 226 (1994) 1021–1024.
[2] K. Drlica, Understanding DNA and Gene Cloning: A Guide for the Curious, Wiley, New York, 1992.
[3] T. Head, Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, Bull. Math. Biol. 49 (1987) 737–759.
[4] H.R. Lewis, C.H. Papadimitriou, Elements of the Theory of Computation, Prentice-Hall, Upper Saddle River, 1987.
[5] New England BioLabs Catalog, 1998/1999.
[6] G. Paun, G. Rozenberg, A. Salomaa, DNA Computing: New Computing Paradigms, Springer, Berlin, Heidelberg, 1998.